

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-211198

(43)Date of publication of application : 02.08.2000

(51)Int.Cl.

B41J 5/30  
G06F 3/12

(21)Application number : 11-019386

(71)Applicant : OKI DATA CORP  
OKI DATA SYSTEMS CO LTD

(22)Date of filing : 28.01.1999

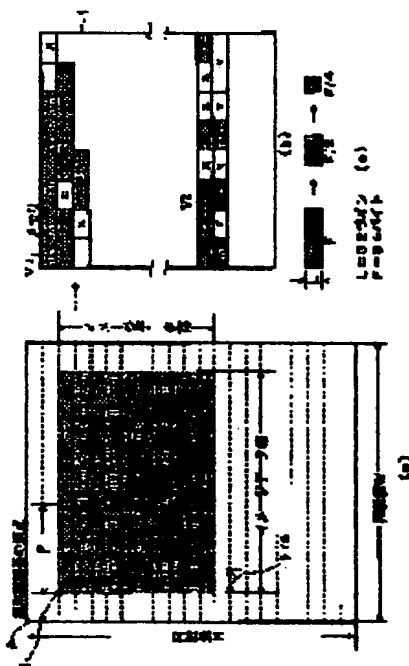
(72)Inventor : SHIMAOKA AKIRA

## (54) METHOD FOR MANAGING IMAGE DATA

## (57)Abstract:

**PROBLEM TO BE SOLVED:** To effectively utilize a memory area by obtaining memory areas continuous on a memory for every block of a second size reduced from a size of the block when it is impossible to obtain a continuous memory area on the memory for storing blocks of a first size.

**SOLUTION:** A block consisting of a plurality of lines of a first size F is set. Image data is stored to memory areas V1 continuous on a memory 4 for every block. If the image data cannot be stored, a process is repeated whereby a size of the block is reduced to a second size F/2 and the image data is stored for every block of the second size. For example, a process of securing the memory data of 64 bytes by 32 lines, namely, approximately 2 kilo bytes as one unit is repeated. If the process is impossible, the memory of (32 bytes × 32 lines), i.e., one kilo bytes is obtained in the middle of the process. Accordingly, the image data can be stored integrally together as much as possible in the memory and a manage amount of data can be minimized.



## LEGAL STATUS

[Date of request for examination]

20.01.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号  
特開2000-211198  
(P2000-211198A)

(43) 公開日 平成12年8月2日(2000.8.2)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	メモリー(参考)
B 4 1 J 5/30		B 4 1 J 5/30	Z 2 C 0 8 7
G 0 6 F 3/12		G 0 6 F 3/12	B 5 B 0 2 1
			9 A 0 0 1

審査請求 未請求 請求項の数4 O L (全 15 頁)

(21) 出願番号 特願平11-19386

(22) 出願日 平成11年1月28日(1999.1.28)

(71) 出願人 591044164  
株式会社沖データ  
東京都港区芝浦四丁目11番地22号  
(71) 出願人 594202361  
株式会社沖データシステムズ  
福島県福島市庄野字立田1番地1  
(72) 発明者 嶋岡 章  
福島県福島市庄野字立田1番地1 株式会  
社沖データシステムズ内  
(74) 代理人 100082050  
弁理士 佐藤 幸男 (外1名)

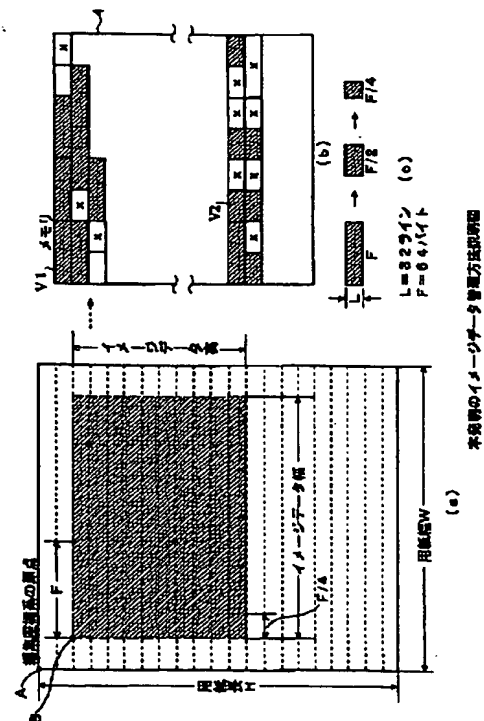
最終頁に続く

(54) 【発明の名称】 イメージデータの管理方法

(57) 【要約】

【解決手段】 上位装置から受信したイメージデータは、予め第1のサイズFの2以上のラインから成るブロックに分割する。各ブロックには印刷されるページ中の印刷位置情報を含む管理情報を用意する。このサイズのブロック毎にメモリ4への格納を繰り返し、記憶領域が見当たらないときは、ブロックのサイズを2分の1にして記憶領域を取得し、格納動作を繰り返す。こうして1ページ分のイメージデータをメモリ4に記憶し、印刷動作を実行する。

【効果】 1ラスタライン毎でなく、所定のサイズの2以上のラインから成るブロックに分割してイメージデータをメモリ4に記憶するので、全体として管理領域のデータ量を減少させ、メモリの使用効率が向上する。



【特許請求の範囲】

【請求項 1】 メモリ上に 1 ページ分のイメージデータを記憶して印刷動作を実行する場合に、

前記イメージデータを一定の第 1 のサイズの 2 以上のラインから成るブロックに分割し、

前記各ブロックに対して、前記ページ中の印刷位置情報を含む管理情報を用意して、

この管理情報を前記メモリに格納するとともに、

前記第 1 のサイズのブロック毎に、前記イメージデータを前記メモリ上の連続した記憶領域を取得して格納し、

前記 1 ページ分のイメージデータ全体を記憶するまで、その動作を繰り返し、

前記第 1 のサイズのブロックのイメージデータを格納するための、前記メモリ上の連続した記憶領域を取得できなくなったときは、前記ブロックのサイズを縮小して第 2 のサイズとし、

前記第 2 のサイズのブロック毎に、前記イメージデータを前記メモリ上の連続した記憶領域を取得して格納し、

前記 1 ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【請求項 2】 請求項 1 に記載のイメージデータの管理方法において、

前記第 2 のサイズのブロックのイメージデータを格納するための、前記メモリ上の連続した記憶領域を取得できなくなったときは、さらに、前記ブロックのサイズを縮小して第 3 のサイズとし、

前記第 3 のサイズのブロック毎に、前記イメージデータを前記メモリ上の連続した記憶領域を取得して格納し、

前記 1 ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【請求項 3】 請求項 1 に記載のイメージデータの管理方法において、

前記各ブロックのイメージデータをメモリに格納する処理を一時中断して、

前記メモリから各ブロックのイメージデータをブロック毎に読み出して圧縮し、メモリに再格納することを特徴とするイメージデータの管理方法。

【請求項 4】 請求項 1 に記載のイメージデータの管理方法において、

前記各ブロックのイメージデータをメモリに格納する処理を一時中断して、

前記メモリから各ブロックのイメージデータを読み出し、複数のブロックのイメージデータを結合してから圧縮して、メモリに再格納することを特徴とするイメージデータの管理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、ページプリンタに

おけるイメージデータの管理方法に関する。

【0002】

【従来の技術】 ページプリンタは、上位装置、例えばパーソナルコンピュータ等から印刷データを受信して印刷をする。通常、ページプリンタでは、装置内部のメモリに 1 ページ分のデータを上位装置から受信した後、イメージデータへの変換処理を行う。1 ページ分の印刷データを編集し、その変換処理が完了すると、印刷動作を開始する。印刷動作の開始と共にメモリ上に保持したイメージデータを順番に読み出し、ラスタバッファに展開する。ラスタバッファには、2 値化されたイメージデータが一時格納され、プリントエンジンに転送される。

【0003】 こうしたページプリンタの印刷データ記述言語としては、HP-PLC 言語が挙げられる。この記述言語によれば、印刷データをラスタライン単位で上位装置からページプリンタに転送する。ラスタラインというのは、主走査方向即ち水平方向の長さが任意の長さであって、副走査方向即ち垂直方向の高さが 1 ドットの、ビットストリーム状データのことをいう。メモリには、このラスタライン単位でイメージデータが書き込まれて記憶される。印刷データが L 本のラスタラインから成り、水平方向の長さ K が全て同一の場合には、長さが K で高さが L の矩形データとして、一括してメモリに書き込まれる場合もある。

【0004】

【発明が解決しようとする課題】 ところで、上記のような従来の技術には次のような解決すべき課題があった。メモリ上に 1 ラスタラインのイメージデータを記憶するためには、その印刷位置、長さ、高さ、解像度等の情報を含む管理情報をメモリに共に記憶する必要がある。解像度が 600 dpi (ドット/インチ) の印刷装置で A4 判の用紙全体にイメージデータを印刷する場合、6000 ラスタライン以上のイメージデータがメモリに格納される。これらのデータを個別に管理する場合、1 ラインで 20 バイト程度の管理情報が必要になるから、全体で約 120 キロバイトの管理情報記憶領域が必要になる。

【0005】 標準的なページプリンタでは、2 メガバイトの RAM (ランダム・アクセス・メモリ) が実装されている。この 2 メガバイトのメモリ空間のうち 0.5 メガバイト分は、ラスタバッファや受信バッファ、システムワーク等のために使用される。従って、運用状態で使用可能なメモリ空間は 1.5 メガバイト程度になる。このことを考慮すると、上記のような管理情報を記憶するために無視できない程度の記憶領域を消費するという問題があった。記憶領域の消費量を低減するためにデータを圧縮することも考えられる。データの圧縮は、メモリに 1 ラスタラインのデータを書き込む直前に、そのつど行われる。しかしながら、ラスタライン単位での圧縮では、圧縮効率が低いという問題がある。

全部  
読み出す

【0006】水平方向の長さがKで高さがLの矩形データとして、一括してメモリに書き込まれる場合には、管理情報も少なくなり、圧縮効率も高まる。しかしながら、このようなイメージデータの生成は上位装置のプリンタドライバに依存する。プリンタドライバの生成するデータは、上位装置上で動作するアプリケーション自体のデータ管理方法や、プリンタドライバとのデータ受け渡し方法に依存する。従って、プリンタ側では、印刷データの形式指定をすることができないから、プリンタ側でメモリを節約するための制御を自発的に行うことができない。

#### 【0007】

【課題を解決するための手段】本発明は以上の点を解決するため次の構成を採用する。

〈構成1〉メモリ上に1ページ分のイメージデータを記憶して印刷動作を実行する場合に、上記イメージデータを一定の第1のサイズの2以上のラインから成るブロックに分割し、上記各ブロックに対して、上記ページ中の印刷位置情報を含む管理情報を用意して、この管理情報を上記メモリに格納するとともに、上記第1のサイズのブロック毎に、上記イメージデータを上記メモリ上の連続した記憶領域を取得して格納し、上記1ページ分のイメージデータ全体を記憶するまで、その動作を繰り返し、上記第1のサイズのブロックのイメージデータを格納するための、上記メモリ上の連続した記憶領域を取得できなくなったときは、上記ブロックのサイズを縮小して第2のサイズとし、上記第2のサイズのブロック毎に、上記イメージデータを上記メモリ上の連続した記憶領域を取得して格納し、上記1ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【0008】〈構成2〉構成1に記載のイメージデータの管理方法において、上記第2のサイズのブロックのイメージデータを格納するための、上記メモリ上の連続した記憶領域を取得できなくなったときは、さらに、上記ブロックのサイズを縮小して第3のサイズとし、上記第3のサイズのブロック毎に、上記イメージデータを上記メモリ上の連続した記憶領域を取得して格納し、上記1ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【0009】〈構成3〉構成1に記載のイメージデータの管理方法において、上記各ブロックのイメージデータをメモリに格納する処理を一時中断して、上記メモリから各ブロックのイメージデータをブロック毎に読み出して圧縮し、メモリに再格納することを特徴とするイメージデータの管理方法。

【0010】〈構成4〉構成1に記載のイメージデータの管理方法において、上記各ブロックのイメージデータをメモリに格納する処理を一時中断して、上記メモリか

ら各ブロックのイメージデータを読み出し、複数のブロックのイメージデータを結合してから圧縮して、メモリに再格納することを特徴とするイメージデータの管理方法。

#### 【0011】

【発明の実施の形態】以下、本発明の実施の形態を具体例を用いて説明する。

〈具体例1〉図1は、本発明のイメージデータ管理方法の説明図である。この図を用いて、まず本発明の概略を説明する。図の(a)には、印刷する用紙のアウトラインと、ここに印刷されるイメージデータ(ハッチングを付した部分)の関係を示した。印刷される用紙は、用紙幅がW、用紙長がHである。編集座標系の原点はAである。印刷されるイメージデータは、ここでは図の破線で囲んだブロック単位で取り扱う。

【0012】図の左上端にあるブロックは、用紙幅Wの方向のサイズがFバイト、用紙長Hの方向のサイズがLドット、即ち長さFでL本のラインにより構成される。この左上端にあるブロックのイメージデータは、その左上隅の点Bの座標を印刷位置情報とする。図示しない管理情報には、このような印刷位置情報が含まれる。図の(b)に、メモリ4のイメージデータを格納するための記憶領域を示した。管理情報は同じメモリ4の別の領域に記憶されているものとする。メモリ4には、イメージデータがブロック毎に記憶される。

【0013】このメモリ4は、図(b)の左上から右下に向かって、順にアドレスが設定されているものとする。(b)に示すV1は、(a)の左上端にあるブロックのイメージデータを格納した記憶領域である。その後も同一サイズの記憶領域に、順に対応するブロックのイメージデータが記憶されていく。一方、メモリ4上には、イメージデータ以外に、例えばダウンロードフォントといった参照データも記憶される。イメージデータを記憶した領域には、1ページの印刷が終わると次のページのイメージデータが上書きされる。しかし、ダウンロードフォントのような参照データは、例えば全てのページの印刷が終了するまで、保持しておかなければならない。これらの参照データがメモリ中に散在すると、必ずしも連続した領域に上記のサイズのブロックのイメージデータを格納することができなくなる。

【0014】そこで、この場合には、ブロックのサイズを、最初のFから2分のFにして、記憶領域を探す。図の(c)には、このように、ブロックのサイズを縮小する例を示した。ライン数Lはそのままである。こうして、メモリ4上にサイズが2分のFのブロックのイメージデータを格納できるような連続した記憶領域を取得できれば、そのブロックを格納する。サイズが2分のFのイメージデータを格納した最初の記憶領域を、図中V2とした。メモリ4上には、印刷開始前に、少なくとも1ページ分のイメージデータが記憶される必要がある。従

って、1ページ分のイメージデータを格納するまで、このような処理を繰り返す。もし、2分のFに縮小したブロックでも連続した記憶領域を取得できない場合には、4分のFのサイズのブロックとし、同様の処理を繰り返す。

【0015】(c)には、例えば、図に示すように、ブロックの第1のサイズFを64バイト、ライン数を32ラインとしたものを示した。このブロックを縮小する場合、第2のサイズを32バイト、第3のサイズを16バイトというように選定した。2分の1ずつ縮小したの

は、コンピュータの処理単位を考慮したためである。管理データは、このようなブロック単位で生成され、メモリの所定の場所に記憶される。

【0016】いずれの場合においても、1ブロックのデータ量が1ラスタラインのデータ量よりも十分に大きければ、従来のように1ラスタライン分ずつ管理データを生成してメモリ4に記憶していく場合に比べて、管理データを減少させることが可能になる。従って、全体として、管理データを記憶すべき記憶領域の節約が可能になる。以上が本発明の方法の概略である。

【0017】図2には、用紙とイメージデータの関係説明図を示す。(a)は、図1に示した通りの用紙で、ここには既に説明した用紙幅W方向のサイズがF、用紙長H方向のサイズがLの、1つのブロックを示した。

(b)には、従来の管理情報と、本発明の管理情報との関係を示す。

【0018】図に示すように、従来は1本のラスタラインRに対し、それぞれ管理情報Dを生成してメモリに記憶するようにしていた。本発明では、2本以上のラインから構成されたブロックVVに対し、管理情報DDを付

加する。

【0019】例えば、長さ600バイトのラスタラインには20バイト程度の管理情報が付加される。一方、上記ブロックのサイズFを62バイトとし、32ライン分について、管理情報DDを付加するとすれば、その管理情報の記憶領域を次のように節約することができる。例えば1ページ6000ラスタラインのイメージデータとすれば、従来、6000個の管理情報の記憶領域として120キロバイトを使用していたが、この発明では例えば1800個の管理情報の記憶領域として36キロバイトを用意すればよい。

【0020】図3には、管理情報の構造説明図を示す。上記のような管理情報は、例えばこの図に示すように、2バイトのチェーンポインタ(Chain pointer)と、1バイトのフォーマットインフォメーション(Format information)と、1バイトのプリントポジション(Print position)Xと、1バイトのプリントポジション(Print position)Yと、1バイトの高さ(Height)、1バイトの幅(Width)、合わせて1バイトの解像度(re)、2バイトのイメージデータアドレス(Image data Address)により構成され

る。

【0021】この例では、これらの管理情報のすぐ後ろに64×32バイトのイメージデータが配置されている例を示す。チェーンポインタは、メモリ開放のための管理ポインタで、他の管理情報との続き具合を示す例えば次の管理情報のアドレスが格納される。フォーマットインフォメーションは、データ種別、データバウンダリ、アドレスバウンダリ等の情報を含む部分である。プリントポジションX、Yは、それぞれ印刷位置のX座標とY座標を示す。例えばブロックの左上隅の座標を示す。

【0022】座標やサイズは、例えば600dpiの解像度の場合、600分の1インチ単位で表示される。その後にイメージデータの高さと幅が表示される。また、解像度は、水平方向即ちX方向の解像度と垂直方向即ちY方向のものを示す。これらは、それぞれ300dpiあるいは600dpiといった形式で表示される。イメージデータアドレスは、この管理情報で管理されるイメージデータの先頭アドレスを表示する部分である。

【0023】このように、管理情報は、ラスタライン1本分のものも、ブロック1個分のものも、表示する内容はほとんど変わらない。ラスタラインの場合には高さ情報が不要な程度である。故に、こうした管理情報の量をブロック単位で取り扱うことによって、メモリ消費量を減少させることができるのである。

【0024】図4には、プリンタの主要部ブロック図を示す。図において、受信処理部1は、プリンタと他の上位装置等とのインタフェースである。この受信処理部1の受信したデータを格納するために、受信バッファ2が設けられる。また、これまで説明した要領でメモリ4上にイメージデータを記憶するために、編集処理部3が設けられている。メモリ4に格納されたイメージデータは編集処理部3を通じて展開処理部5に読み出され、展開されてラスタバッファ6に転送されるよう構成されている。なお、このラスタバッファ6に格納されたイメージデータは、図示しないプリントエンジンに転送されて印刷される。

【0025】編集処理部3には、受信データ解析部11、編集環境設定処理部12、イメージデータ処理部13、図形データ処理部14、文字データ処理部15、他データ処理部16、ディスプレイリスト生成処理部17、メモリ管理部18、イメージデータディスプレイリスト生成処理部19及びデータ圧縮処理部20が設けられている。なお、データ圧縮処理部20は、具体例2を実施する場合に利用される部分である。

【0026】図5には、イメージデータ管理動作フローチャートを示す。上記のプリンタは、このフローチャートに従って動作する。まず、上位装置から転送される印刷データは受信処理部1が受信する。受信した印刷データの1ビットがイメージデータの1ドットに相当する場合と、印刷データの2ビットまたは4ビットあるいは8

ビットの複数ビットが、イメージデータ1ドットの階調値を指定する場合とがある。この例では、印刷データ1ビットがイメージデータ1ドットに相当する場合を説明する。印刷データが階調値を指定する場合には、例えばディザ法等により2値のマトリクスに展開処理を行い、全体が2値化された後のイメージデータを使用する。

【0027】また、印刷データは一定の規則に従って圧縮されている場合もある。受信処理部1は、こうした印刷データを一旦受信バッファ2に記憶する。次に、編集処理部3の受信データ解析部11が、受信バッファ2から印刷データを取り出し、その内容を解析する。印刷データは、従来どおりラスタライン単位で受信されるものとする。

【0028】ここで、受信データ解析部11は、各取り出した印刷データの印刷位置を算出する。印刷データの属性データには、印刷位置がX座標とY座標とで表示されている。印刷位置はそのラスタラインの先頭のビットを表示する位置である。水平方向の印刷位置は、例えば(X座標/16)の商の整数部分とする。その端数は実印刷位置との差として保持し、展開時に補正する。垂直方向の印刷位置は(Y座標/32)の商の整数部分とする。端数はメモリへの格納位置補正に使用する。

【0029】即ち、ここでは、水平方向のイメージデータの位置管理を16ドット単位で行う。また、垂直方向のイメージデータの位置管理を32ライン単位で行う。

(X座標/16)の商の整数部分をとると、実際に指定されている印刷位置よりも、“1”から“15”ドットの範囲で水平方向に位置ずれを生じる。これは、記憶領域を取得するきっかけとなったブロックの最初のラインの印刷位置(X座標)を基準に補正すればよい。垂直方向も同様である。これらの計算結果を利用して、処理対象となったラスタラインを分割して得たイメージデータを、対応するブロックのために取得した記憶領域に順に位置補正をしながら格納していく。

【0030】次に、ステップS2において、印刷データが圧縮されているかどうかを判断する。圧縮されていればステップS3に進み、圧縮アルゴリズムを使用して、印刷データを伸長する。この段階で全ての印刷データは2値の加工されていないイメージデータに変換された。ステップS4では、図4に示すイメージデータ処理部13が、入力するイメージデータを既に説明した一定のサイズに分割する。即ち、1ラスタライン分のイメージデータの先頭から64バイト分を分割して取り出す。

【0031】次に、ステップS5において、記憶領域取得の可否を決定する。ここで記憶領域取得の可否を決定するのは、次のような理由による。受信データは、ラスタライン毎に受信される。そして、例えば最初のラスタラインを受信したとき、その先頭から64バイト分を格納するために、メモリ上に所定の連続した記憶領域を取得する。このとき取得する記憶領域は、64バイト×3

2ライン分である。そのラスタラインのサイズ即ち図1のイメージデータ幅と表示した部分のサイズが64バイト以上あれば、残りの部分についても64バイトずつに分割し、それぞれ別々の記憶領域を64バイト×32ライン分ずつ取得する。

【0032】次に新たなラスタラインを受信すると、その先頭の64バイト分を、直前のラスタラインの先頭の64バイト分のイメージデータに続けて格納する。この場合には、既に記憶領域が取得されているから、新たに記憶領域を取得する必要はない。この処理を32ライン分続ければ、サイズが64バイトで32ライン分の例えば3個のブロックが、1ブロック分ずつメモリ上の連続した記憶領域に格納される。次のラスタラインが受信されたときは、新たに3個のブロックのための記憶領域を取得する。即ち、32ラスタラインごとに新たな記憶領域の取得を行うので、ステップS5を設けたのである。

【0033】図6には、こうした記憶領域取得動作のフローチャートを示す。まず、ステップS51において、記憶領域が取得済みかどうかを判断する。ブロックの最初のラインを記憶する場合には、記憶領域の取得が必要であるからステップS52に進み、「取得要」という判断結果をフラグ等で表示して、図5のステップS6に進む。ブロックの第2番目以下のラインを記憶する場合には、記憶領域が取得済みであるからステップS51からステップS53に進む。

【0034】そして、ここで、これから格納しようとするイメージデータが、既に格納済みのイメージデータと水平方向の位置が重複するかどうかを判断する。イメージデータの先頭の位置が一致すれば、両方とも64バイトのサイズであって、そのまま確保した記憶領域に格納できるから、ステップS57で「取得不要」というフラグを立てて、ステップS56に進む。もし、イメージデータの先頭位置が一致しないと、64バイトのイメージデータであっても、先頭位置のシフト分だけ、サイズが実質的に大きくなる。即ち、確保しておいた64バイト分の記憶領域に格納できない。

【0035】そこで、その場合には、イメージでのサイズをシフト分だけ短くするように変更する。ステップS55はそのための処理である。ステップS54はステップS57と同一の処理である。その後ステップS56で、水平方向と垂直方向のデータシフト量を算出して、そのイメージデータを格納すべき位置を求め、次の処理に進む。

【0036】再び図5に戻って、ステップS6では、上記の記憶領域取得可否の決定に基づいて、処理を分岐する。記憶領域の取得が必要でなければステップS10に進み、図6のステップS56の処理結果に従ってメモリの所定位置にイメージデータを格納する。次にステップS11において、1ラスタライン分のイメージデータのうち分割した残りのイメージデータがあれば、そのサイ

ズをもとのサイズから64バイト分差し引いたサイズに更新する。次のステップS12において、全てのイメージデータの格納が完了したかどうかを判断する。1ページ分のイメージデータの処理を連続して行うためである。

【0037】全てのイメージデータの格納が完了していなければステップS4に進み、残りのイメージデータについて、先頭から64バイト分を分割し、これまで説明した処理を行う。残りのイメージデータが64バイトに満たない場合にも、1ブロックのサイズは64バイトとする。後から1ブロックのサイズが変更された場合には、それに従う。また、ステップS5において記憶領域の取得が必要と判断されたときは、ステップS6からステップS7に進み、メモリ4上に1ブロック分の記憶領域を取得する。この処理は、図4に示すメモリ管理部18が行う。この場合、メモリ4上に1ブロック分のイメージデータを格納できる連続した記憶領域を探して、その先頭アドレスを得る。その後ステップS7からステップS8に進み、記憶領域を取得できたかどうかを判断し、取得できた場合にはステップS10に進み、メモリにそのブロックの先頭のラインのイメージデータを格納する。

【0038】ここで、連続した記憶領域が取得できない場合には、ステップS8からステップS9に進み、ブロックサイズを2分の1にする。即ち、64バイト32ライン分のブロックサイズを32バイト32ライン分にして、改めてメモリ4をサーチし、連続した記憶領域を取得する。以上のような処理を、ステップS12で、1ラスタライン分のイメージデータの格納が完了するまで繰り返す。改ページ等のコマンドが検出されるまで、各ラスタラインのイメージデータを以上のような処理を繰り返すことにより、メモリに格納する。

【0039】改ページコマンドを検出すると、メモリ4へのイメージデータの格納が終了する。その後、メモリ4中のイメージデータは、イメージデータディスプレイリスト生成処理部19を通じて展開処理部5に読み出される。展開処理部5は、イメージデータを順次読み出してラスタバッファ6を通じて、図示しないプリントエンジンに転送する。上記のように、上位装置から1ラスタラインずつイメージデータを受信して、64バイトのサイズに分割し、それをメモリに格納する作業を32回繰り返して1ブロック分のイメージデータの格納を終えるが、そのデータは切れ目なく連続しており、その後は先頭アドレスとブロックの大きさを管理情報により表示しておけば、一括してメモリから読み出すことができる。

【0040】上記のように一定のサイズ(第1のサイズ)の複数のラインから成るブロックを設定し、イメージデータをブロック毎にメモリ上の連続した記憶領域に格納し、それで格納できない場合にはブロックのサイズを縮小して、第2のサイズのブロック毎にイメージデー

タを格納するという処理を繰り返せば、メモリ中のデータの再配置処理を行うことなく、記憶領域を有効に利用することができる。即ち、既に説明したように、メモリ上には様々な保持しておかなければならないデータが散在することがあり、これらの存在によって、1ブロック分のイメージデータを連続して格納することができなくなる。この場合、これらを再配置させる処理を行えばよいが、印刷中にこのような処理を行えば、印刷速度が低下する。

【0041】従って、図1(a)に示したように、例えば上記のように64バイトで32ライン、即ち約2キロバイトを1単位として記憶領域を確保する処理を繰り返し、これが不可能ならば途中から32バイト×32ライン即ち1キロバイトのメモリ取得を行うようにする。これでも連続した記憶領域の取得が不可能ならば、後半は16バイト×32ライン即ち0.5キロバイトの記憶領域取得を行うようにして1ページ分のイメージデータ格納処理を終える。こうすれば、可能な限りイメージデータをまとめてメモリに格納し、管理データ量を最小限にすることができる。

【0042】ところで、上記のようなメモリに格納されたイメージデータは、印刷開始と同時にラスタバッファ6を通じてプリントエンジンに転送される。この場合のラスタバッファ6上へのイメージデータ展開速度は、用紙走行速度を考慮して定められる。即ち、一定時間内にラスタバッファ6への展開が完了しない場合には、いわゆるプリントオーバーランという現象が発生する。プリントエンジンは印刷用紙を搬送するが、必要なイメージデータの転送が間に合わず、白紙のまま印刷をしてしまう現象である。データの管理単位を1ラスタライン毎に設定しておく、メモリ4から1ラスタライン毎にイメージデータが読み出されてラスタバッファ6へのデータ転送が行われる。

【0043】これに対して、本発明のように、ブロック毎のイメージデータ管理を行うと、ラスタラインよりもデータ量が大きくなるように設定したブロック毎にイメージデータが読み出されて、ラスタバッファ6へのデータ転送が行われる。故に、イメージデータを展開し、ラスタバッファに転送する際の処理時間が短縮され、プリントオーバーランを防止する効果がある。

【0044】例えば、いずれもRISC-CPUを使用した2台のプリンタA、Bであって、一方は8ppm(ページ/分)、他方は16ppmの印刷速度のものをシミュレーションにより比較して見る。プリンタAは、制御クロック周波数25メガヘルツ、Iキャッシュ(制御データ用キャッシュ)が4キロバイト、Dキャッシュ(データ用キャッシュ)が1キロバイトとする。プリンタBは、制御クロック周波数32メガヘルツ、Iキャッシュ3キロバイト、Dキャッシュ1キロバイトとする。

【0045】文字データを含むビットマップデータのラ

スタバッファへの展開に要する前処理時間は、プリンタ A、B 共に1016サイクルであった。いずれのプリンタもラスタバッファは192ラインの容量を持ち、用紙の走行速度はプリンタ Aが2インチ/秒、プリンタ Bは4インチ/秒であった。タスクの切り換え、割り込み、ビデオDMAのバス占有等、展開処理以外の処理に要する時間を全体の処理の20%とすると、プリンタの解像度が600dpiの場合、192ラインの展開に許される時間は次のようになる。

プリンタ A:  $(192 \div (2 \times 600)) \times 0.8 = 0.128$  10

プリンタ B:  $(192 \div (4 \times 600)) \times 0.8 = 0.064$

【0046】即ち、プリンタ Aは128ミリ秒、プリンタ Bは64ミリ秒の展開可能時間となる。この時間内に展開処理を終了すれば、オーバーランにならない。イメージデータ展開の前処理が1ラスタライン当たり1016サイクルであって、192ラスタラインの前処理には195702サイクルが必要となる。従って、それぞれのクロック周波数を考慮した展開可能時間に占める前処理時間の割合は次のようになる。プリンタ A:  $(195702 \text{ サイクル} \div 25 \text{ メガヘルツ}) \div 128 \text{ ミリ秒} = 0.0611$  プリンタ B:  $(195702 \text{ サイクル} \div 32 \text{ メガヘルツ}) \div 64 \text{ ミリ秒} = 0.955$  20

【0047】即ち、プリンタ Aは6.11%、プリンタ Bは9.55%となる。即ち、CPUの搭載周波数と用紙の走行速度との関係によってある程度変化するが、この前処理に要する時間の割合が小さいほど、イメージデータと他のデータの展開に許される時間が増加する。従って、本発明のように、ブロック単位の処理を行い、ブロック毎に管理情報を用意することによって、展開に時間のかかるようなイメージデータを処理したとしても、プリントオーバーラン発生率の低下を図ることができる。 30

【0048】〈具体例1の効果〉以上のように、ブロック単位でメモリ上に連続した記憶領域を取得し、その動作を繰り返し、第1のサイズのブロックを格納するためのメモリ上の連続した記憶領域を取得できなくなったとき、ブロックのサイズを縮小して第2のサイズとし、再び第2のサイズのブロック毎にメモリ上の連続した記憶領域を取得するといった方法で2以上のラインから成る一定のサイズのブロックでメモリを管理すると、管理情報の減少によってメモリ領域を有効に利用できる。更に、ラスタラインよりもデータ量が大きくなるように設定したブロックを用いるので、ラスタバッファへの転送処理のための時間を減少させ、プリントオーバーランの発生を低減できる。 40

【0049】〈具体例2〉上記の例では、一定の容量のメモリに格納する管理データの量を減少させて、メモリを有効に活用する方法を示した。しかしながら、有限の 50

メモリにいかに大量のイメージデータを記憶することができるかが、プリンタ等の製品の性能を評価する指標になっている。従って、イメージデータをメモリに効率よく格納するだけでなく、よく知られたデータ圧縮を併用することにより、上記の本発明を一層効果的に利用することができる。

【0050】例えば、上記のように3段階のサイズを設定し、最初64バイト、次は32バイト、その次は16バイトのサイズでブロックを設定して連続する領域を探しても、連続してイメージデータを記憶する領域が見当たらないような場合には、メモリフルという状態が発生する。1ブロックのデータ量を1ラスタラインより小さくしてしまえば、イメージデータをブロックにまとめた効果がない。そこで、この具体例では、メモリに格納されている全てのデータをブロック単位で順番に読み出し、それぞれよく知られたアルゴリズムを用いて圧縮する。これによって、メモリの記憶領域を増大させる。

【0051】図7には、ファイル圧縮率の説明図を示す。ここでは、例えば様々なデータ形式のファイルを120種類ほど用意し、それぞれ2種類の圧縮方式を採用し、元のデータに対し圧縮後のデータ数（単位はバイト）がどの程度になったか、その圧縮率はどの程度かという結果を示している。圧縮元のデータ数は、全ファイル、幅600バイト、高さ6000ラスタライン固定の3,840,000バイトである。例えば、ファイル番号が1番のファイルは、図に示すように、圧縮方式Aでは、圧縮後のファイルサイズは252,221バイトであって、6.56%の圧縮率となる。データの形式やファイルの形式や圧縮方式によって、圧縮率が様々な範囲に変化することがわかる。 50

【0052】図8には、このようなファイル圧縮率に着目して、2種の圧縮方式を用いた場合の頻度表を図示した。例えば、図に示す圧縮方式Aでは、圧縮率が5%未満のものが最も頻度が高く、圧縮率が10~15%未満のものが27ファイルある。また、圧縮率が45~50%未満のものも1%存在する。圧縮方式Bでも類似した傾向が見られる。こうした結果から、平均的には圧縮前のメモリ消費量の30%程度まで、ファイルサイズを圧縮することができるということが考えられる。

【0053】従って、メモリに格納されたデータをメモリフルを検出した段階あるいはそれ以前の適当なタイミングで圧縮すれば、その時点で、残りのイメージを格納することができる記憶領域を確保できる。図4に示すデータ圧縮処理部20は、このような圧縮処理を行うために用意された部分である。圧縮アルゴリズムについては、従来よく知られた任意の方法で良い。既に説明したように、2MバイトのRAMを実装したページプリンタでは、運用時に使用可能な記憶領域は1.5Mバイト程度である。一方、A4判の用紙全体に600dpiで印刷をする場合、6000ラスタライン以上のイメージデ



ータが必要で、そのデータ量は4Mバイト程度となる。  
上記の圧縮処理を実行すれば、実装メモリの増設が不要になる。

【0054】図9は、具体例2の処理動作フローチャートである。このフローチャートは、具体例1のような処理を実行中に、メモリフル状態が発生したとき、実行される処理を示す。まず、ステップS1において、メモリがイメージデータを保持しているかどうかを判断する。メモリ中にイメージデータが存在していればステップS2に進み、メモリ中の全てのイメージデータをブロック毎に取り出し、順番に圧縮する。次に、ステップS3において、要求のあった記憶領域を取得できるかどうかの判断を行う。

【0055】即ち、圧縮処理を終了した後にメモリの記憶領域に十分な空きが発生すると、具体例1の記憶領域取得処理を続けることができる。記憶領域が取得できればステップS5に進み、その記憶領域を取得して、具体例1のイメージデータ格納処理に進む。記憶領域を取得できなければ圧縮をしても効果がなかったと判断し、ステップS4において、メモリフルという表示等を行い、ページ編集を終了すればよい。

【0056】〈具体例2の効果〉以上のように、任意のタイミングであるいはメモリフルが発生した場合に、メモリ中のデータ圧縮を行うことによって、メモリをより有効に使用することが可能になる。特に、メモリフルを検出した場合のみデータ圧縮を行うことによって、不必要なデータ圧縮処理による処理速度の低下を回避することができる。例えば従来のようにラスタライン単位で上位装置からデータの転送を受ける場合、データを圧縮するモードか圧縮しないモードかをデータの受信開始時に決定して、圧縮が必要ならば全てのデータについて圧縮をする。従って、処理速度の低下を招いていた。一方、本発明では、メモリフルになった時点でメモリ中のデータ圧縮をするから、メモリフルにならなければ、無駄な圧縮はしない。また、ブロック単位でデータを読み出して圧縮するので、ラスタライン単位で読み出すよりも高速に処理できるという効果がある。

【0057】〈具体例3〉上記の例では、メモリに格納されたイメージデータをブロック単位で管理していることに着目し、メモリからブロック単位でイメージデータを取り出し、これを順次圧縮し、メモリに再格納するという方法を採用した。しかしながら、各ブロックは最大2キロバイト程度であって、従来の1ラスタライン単位で管理されていたデータを圧縮するより圧縮率は改善できるが、それ以上の圧縮率は望めない。ところが、圧縮対象となるイメージデータのデータ数と圧縮効率の関係をシミュレーションすると、データ数が増加するほど圧縮率が向上し、圧縮対象データ数が8キロバイトを超える範囲でほぼ飽和することが判明した。

【0058】図10には、このデータサイズ別圧縮率比

較説明図を示す。図に示すように、高さが128ライン、64ライン、32ラインのものであって、幅16バイトのブロック、幅32バイトのブロック、幅64バイトのブロック、幅128バイトのブロック、幅256バイトのブロック、幅512バイトのブロックについて、それぞれデータ圧縮を行うと、その圧縮後のデータ数と圧縮率が様々な結果を示す。(a)は比較的圧縮率が低い場合の例で、(b)は比較的圧縮率が高い場合の例である。

【0059】この結果からみると、幅128バイト〜幅512バイトのブロックサイズの場合は、圧縮データの個々のサイズが8キロバイトを超えている。このような点を考慮すると、1ブロックのデータ数は大きい方がよいといえる。しかしながら、上記のようなメモリへのデータ格納のために連続した記憶領域を確保する処理を伴う場合には、ブロックを適切なサイズに選定することが好ましい。そこで、この具体例3では、データ圧縮の際、いくつかのブロックを読み出し、複数のブロックをまとめて圧縮し、メモリに再格納するという処理を行う。これによって、圧縮率を向上させている。

【0060】図11は、具体例3の動作説明図である。また、図12は、この結合圧縮動作の説明図である。更に、図13は、具体例3の動作フローチャートである。これらを用いて具体例3の動作を順に説明する。

【0061】図11(a)に示すように、これまで説明した通りの用紙のレイアウト中に、(1)-1〜(8)-3及びA1〜C5のブロックが存在し、これらが既に説明した要領でメモリ中に格納されているものとする。図の(1)-1のブロックの最初のラスタラインのイメージデータを受信すると、(1)-2、(1)-3のブロックのイメージデータについて、同時にブロック毎の記憶領域を取得し、既に説明した要領で、1ラスタライン受信する度に、順番に該当する記憶領域にイメージデータを格納する。このような処理が図の右側に示した(1)〜(11)まで繰り返される。

【0062】なお、(9)では、ブロックのサイズが2分の1にされ、これまでと同様の要領でメモリへの格納処理を続ける。ここで、圧縮処理を行う場合には、

(1)に示す3つのブロック即ち(1)-1のブロック、(1)-2のブロック、(1)-3のブロックをまとめてから圧縮を行う。また、図11(b)には、各ブロックのイメージデータのデータ構造を図示した。それぞれ、64バイト×32ライン分のデータが連続した構造をしている。上記の状態でもメモリフルが発生したとする。ここで、つぎのようにしてデータの圧縮を行う。

【0063】図12には、この具体例3における結合圧縮動作を示す。図12(a)に示すように、例えば図11の(1)-1から(1)-3の3つのブロックのイメージデータを結合させ、連続する1つのデータとして圧縮処理をする。(1)-1と(1)-2は64バイトの

サイズのブロックで、(1) - 3は32バイトのサイズのブロックである。これらを結合し、図12に示すように、 $64 \times 2 + 32$ バイトのサイズで32ラインのイメージデータから成るブロックを得る。即ち、サイズ160バイト32ラインのブロックを一括して圧縮対象とする。図12(b)では、例えば、A1~A5のブロックを5個まとめる。それぞれ32バイトのサイズのブロックであるため、これらを5つ結合し、サイズ160バイトで32ラインのブロックとして圧縮を施す。こうすれば、既に説明をしたように、圧縮率がより高まる。

【0064】図13は、上記の処理を説明するための動作フローチャートである。この処理は、具体例2と同様に、メモリフルの状態が発生してから開始する。図に示すように、ステップS1において、未圧縮のイメージデータがあるかどうかをメモリ中で検索し、存在すればステップS2に進み、水平方向に隣接するデータがあるかどうかを判断する。即ち、図12を用いて説明したように、水平方向に隣接するデータがある場合には、これらを全てまとめて結合し、1つのブロックとして圧縮を行うようにする。隣接データが存在する場合にはステップS3において、隣接するデータを結合させて圧縮する。存在しない場合はステップS4に進み、単独のメモリの圧縮を行う。以上の処理によって、圧縮率のより高いデータサイズにした上で圧縮を行う。

【0065】以上のような処理は、印刷する用紙がランドスケープ形式の場合でも可能である。ランドスケープ形式とは、用紙を横に使用する形式で、これまで説明したのは、用紙を縦に使用するポートレイト形式である。図14には、ランドスケープの場合の動作説明図を示す。ランドスケープの場合、例えばこの図に示すように、ブロックを用紙長Hの方向にとる。即ち、用紙長Hの方向にブロックの第1のサイズ即ち64バイトの長さを取り、用紙幅Wの方向にラインを設定してイメージデータを処理する。その他の処理手順についてはこれまでの例と全く同様である。

【0066】図15には、ランドスケープの場合の圧縮処理説明図を示す。図の(a)に示すように、(1) - 1~(8) - 3までの第1のサイズのブロックと、A1~C5までの第2のサイズのブロックのイメージデータをメモリ中に格納している。これが1ページ分のデータとする。ここで、ランドスケープのデータは、メモリから読み出されてラスタバッファ6に転送される場合、90°回転させて、ポートレイト形式のデータに変換される。プリントエンジンはポートレイト形式のデータのみを受け付ける構造にされているからである。図15

(b)には、例えば(1) - 1~(8) - 1のデータを回転させた後の、データを示す。この例では、(1) - 1~(8) - 1のデータを回転させてから結合し、これらをまとめて圧縮する。

【0067】図16には、回転後の結合圧縮動作説明図

を示す。図16(a)に示すように、例えば(1) - 1の64バイト32ラインのブロックが回転されて、サイズ4バイトの512ラインのブロックに変換される。他の(2) - 1~(8) - 1のブロックも同様に回転される。そして、回転後の(1) - 1~(8) - 1の8個のブロックが結合される。こうして16キロバイトの圧縮対象ブロックを生成する。

【0068】(b)は、A1~C5のメモリ空間について、その回転後の結合圧縮を示す動作説明図である。この場合、A1, B1, C1の32ライン32バイトのブロックを回転して4バイト256ラインのブロックに変換する。そして、これら3つのブロックを結合して、4バイト×3のサイズで256ラインのブロックを生成し、圧縮対象を3キロバイトとして圧縮する。このように、回転後、結合するという処理によって、ランドスケープタイプについても同様のデータ圧縮が可能になる。

【0069】図17には、具体例3の回転圧縮の場合の動作フローチャートを示す。この処理も、メモリフルの状態が発生した後で開始する。まず、ステップS1において、未圧縮イメージデータがあるかどうかを判断し、あればステップS2に進み、垂直方向に隣接データがあるかどうかを判断する。そして、垂直方向に隣接するデータがあればステップS3に進み、隣接する個々のメモリの回転を行う。更に、ステップS4に進み、図15、16で説明した要領で、回転後の複数のメモリを結合させて圧縮する。また、垂直方向に隣接するデータがなければステップS5に進み、メモリの回転を行って、回転後のメモリの圧縮を単独で行えばよい。

【0070】図18には、圧縮対象データの分割例説明図を示す。図には、512バイト128ラインのブロックと、これを16バイトサイズで分割した場合、32バイトサイズで分割した場合、64バイトサイズで分割した場合、128バイトサイズで分割した場合、256バイトサイズで分割した場合を図解した。具体例1では64バイトサイズ32ラインのブロックを1単位として処理した。ところが、データの圧縮をする場合には、図10を用いて説明したように、データ数の大きいもののほど圧縮率が高い。そのデータ数を図示比較したものがこの図18である。

【0071】例えば64バイト、高さ32ラインのブロックで分割してイメージデータを管理し、その各ブロックをそのブロック単位で圧縮した場合には、低圧縮率のデータを圧縮する場合、80.0%、高圧縮率のデータを圧縮する場合、29.75%程度までデータ圧縮ができる。

【0072】一方、具体例3のように、また、隣接するブロックを結合させて一括して圧縮処理をすると、低圧縮率のデータを圧縮する場合には、74.18%、平均的なデータを圧縮する場合、21.31%まで圧縮することができる。圧縮後にメモリに約80%の空きが生じ

るわけである。このような結果からブロックをどの程度結合するかを選定してその最適化を図ることができる。

【0073】〈具体例3の効果〉以上のように、ブロック毎にイメージデータを管理し、メモリに格納した場合においても、隣接するブロックを結合して、一括して圧縮をすることにより、圧縮効率が向上し、メモリの使用効率を向上させることができる。

【図面の簡単な説明】

【図1】本発明のイメージデータ管理方法説明図である。

【図2】用紙とイメージデータの関係説明図である。

【図3】管理情報の構造説明図である。

【図4】プリンタの主要部ブロック図である。

【図5】イメージデータ管理動作フローチャートである。

【図6】記憶領域取得動作フローチャートである。

【図7】ファイル圧縮率の説明図（その1）である。 \*

\* 【図8】ファイル圧縮率の説明図（その2）である。

【図9】具体例2の動作フローチャートである。

【図10】データサイズ別圧縮率比較説明図である。

【図11】具体例3の動作説明図である。

【図12】結合圧縮動作の説明図である。

【図13】具体例3の動作フローチャートである。

【図14】ランドスケープの場合の動作説明図である。

【図15】ランドスケープの場合の圧縮処理説明図である。

10 【図16】回転後の結合圧縮動作説明図である。

【図17】具体例3の動作フローチャートである。

【図18】圧縮対象データの分割例説明図である。

【符号の説明】

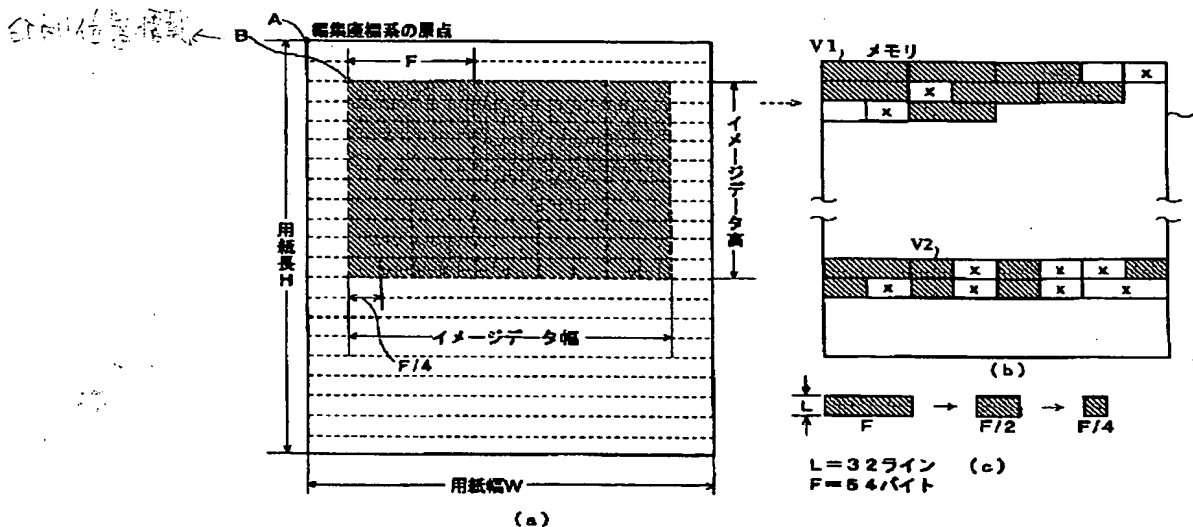
4 メモリ

F 第1のサイズ

F/2 第2のサイズ

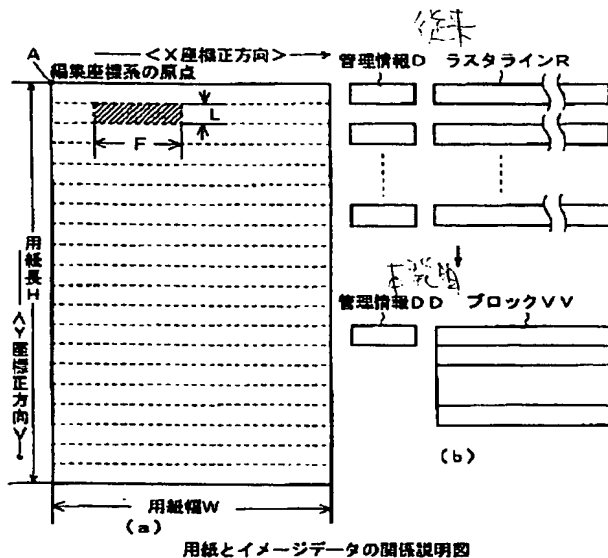
V1, V2 記憶領域

【図1】



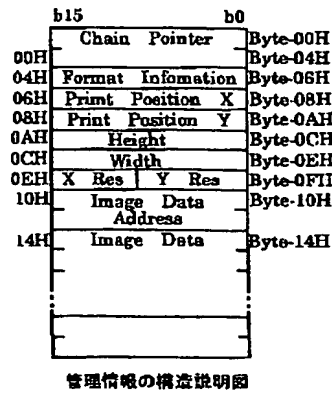
本発明のイメージデータ管理方法説明図

【図2】

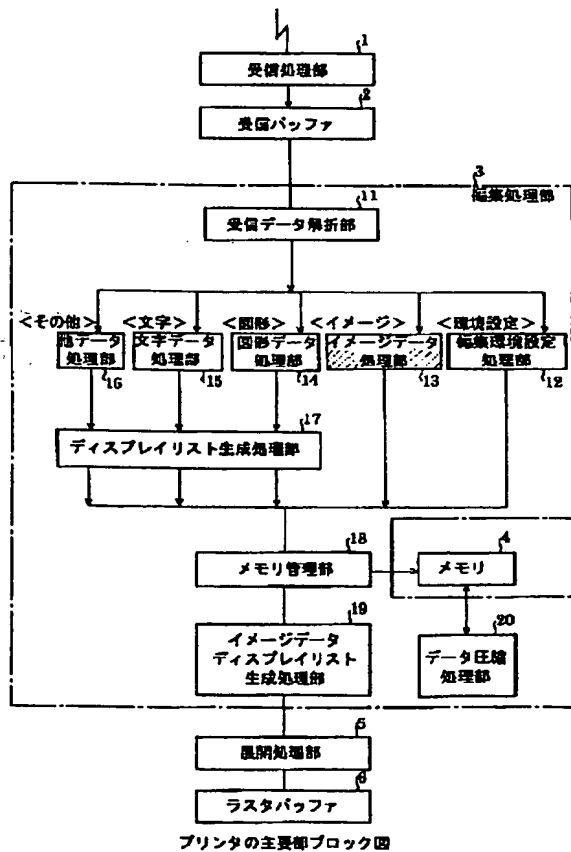


用紙とイメージデータの関係説明図

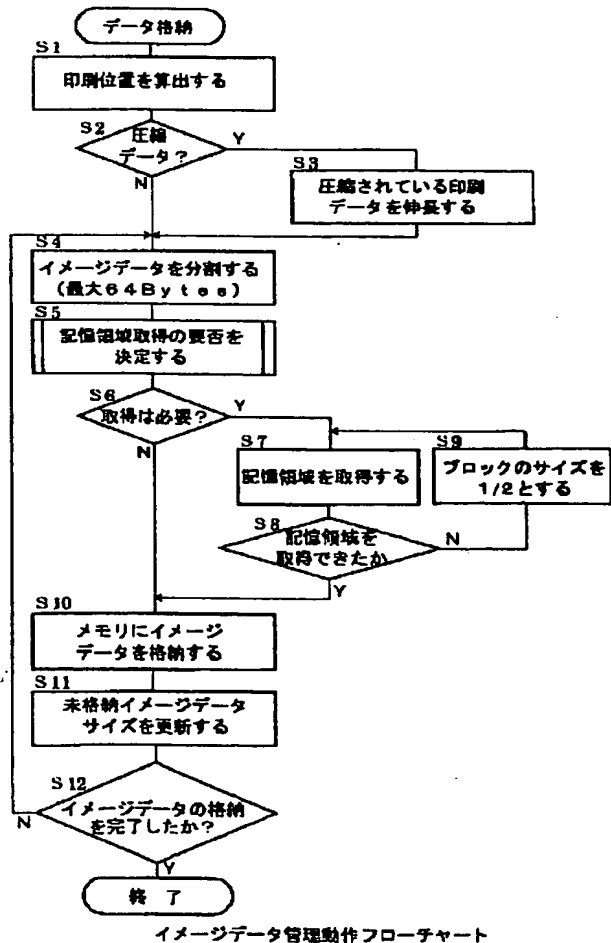
【図3】



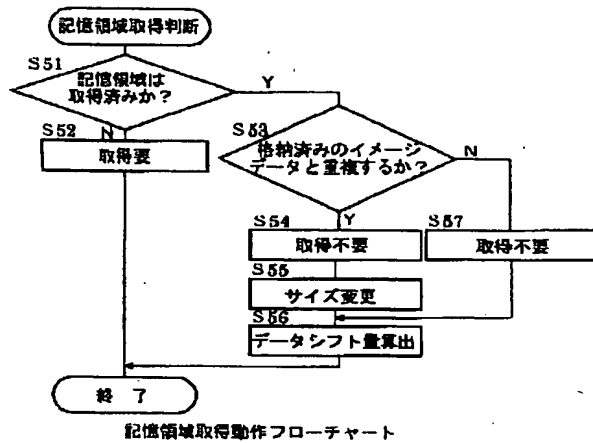
【図4】



【図5】



【図 6】



【図 7】

File No	圧縮後データ		圧縮率	
	圧縮方式(A)	圧縮方式(B)	圧縮方式(A)	圧縮方式(B)
1	262,221	215,817	8.56%	5.62%
2	109,732	82,592	2.86%	2.16%
3	240,276	201,487	6.26%	5.25%
4	98,819	81,453	2.57%	2.12%
5	186,285	149,617	4.86%	3.90%
...				
36	127,986	142,827	3.32%	3.72%
37	437,542	509,460	11.39%	13.27%
38	468,571	548,749	12.20%	14.29%
39	350,102	919,723	9.12%	23.98%
40	327,532	845,617	8.53%	22.03%
...				
119	291,425	275,009	7.59%	7.16%
120	394,888	261,248	7.68%	6.80%

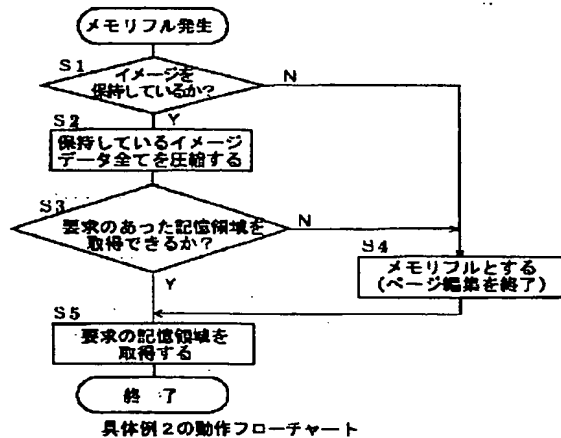
ファイル圧縮率の説明図 (その1)

【図 8】

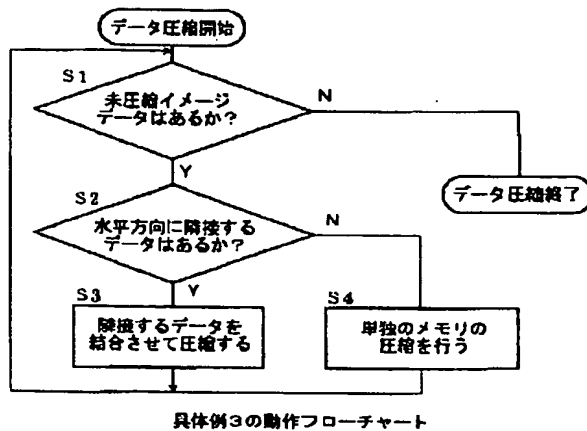
圧縮率	圧縮方式(A)	圧縮方式(B)
～5%未満	33	25
5%～10%未満	39	27
10%～15%未満	27	29
15%～20%未満	8	15
20%～25%未満	3	11
25%～30%未満	3	2
30%～35%未満	2	4
35%～40%未満	1	0
40%～45%未満	1	2
45%～50%未満	1	1
50%～55%未満	1	2
55%～60%未満	0	0
60%～65%未満	0	1
65%～70%未満	0	0
70%～75%未満	1	0
75%～80%未満	0	0
80%～85%未満	0	1
85%～90%未満	0	0
90%～95%未満	0	0
95%～100%	0	0

ファイル圧縮率の説明図 (その2)

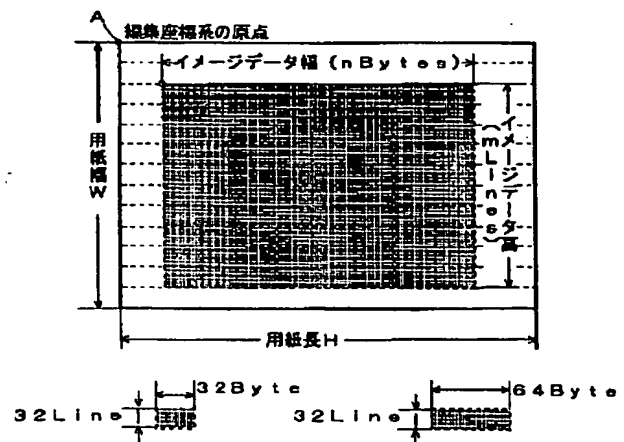
【図 9】



【図 13】



【図 14】



ランドスケープの場合の動作説明図

【図 10】

	圧縮前 データ数	幅:16 Bytes データ数 圧縮率	幅:32 Bytes データ数 圧縮率	幅:64 Bytes データ数 圧縮率
高さ:128 Lines	66,536	64,955 99.13%	57,049 87.05%	52,779 80.53%
高さ: 64 Lines	32,768	32,483 99.13%	28,508 87.00%	26,284 80.21%
高さ: 32 Lines	16,384	16,059 98.02%	14,190 86.61%	13,112 80.03%

	圧縮前 データ数	幅:128 Bytes データ数 圧縮率	幅:256 Bytes データ数 圧縮率	幅:512 Bytes データ数 圧縮率
高さ:128 Lines	66,536	50,435 78.98%	49,092 74.91%	48,362 73.79%
高さ: 64 Lines	32,768	25,127 76.68%	24,424 74.54%	24,049 73.39%
高さ: 32 Lines	16,384	12,608 76.95%	12,323 75.21%	12,153 74.18%

圧縮率低  
(a)

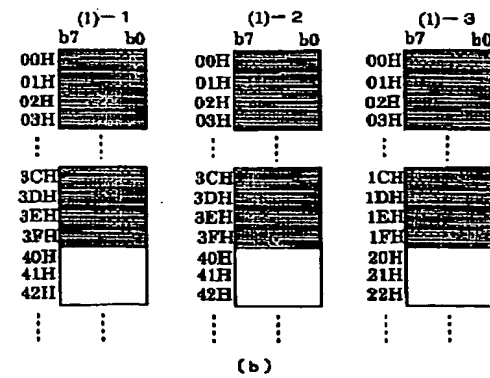
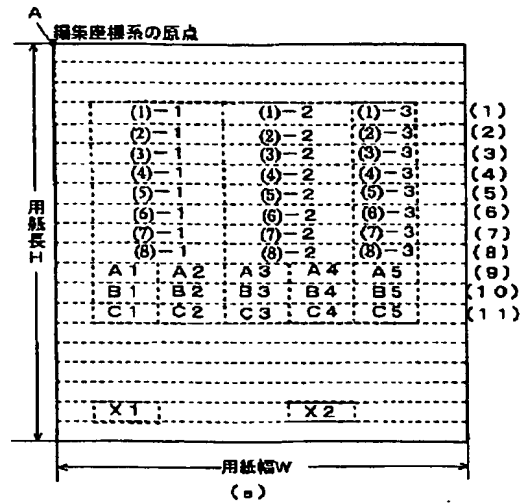
	圧縮前 データ数	幅:16 Bytes データ数 圧縮率	幅:32 Bytes データ数 圧縮率	幅:64 Bytes データ数 圧縮率
高さ:128 Lines	66,536	35,305 53.87%	26,093 39.82%	20,939 31.95%
高さ: 64 Lines	32,768	17,696 54.00%	13,012 39.71%	10,440 31.86%
高さ: 32 Lines	16,384	8,632 52.69%	6,228 38.01%	4,874 29.76%

	圧縮前 データ数	幅:128 Bytes データ数 圧縮率	幅:256 Bytes データ数 圧縮率	幅:512 Bytes データ数 圧縮率
高さ:128 Lines	66,536	18,149 27.69%	16,605 25.34%	15,687 23.94%
高さ: 64 Lines	32,768	8,985 27.42%	8,239 25.14%	7,766 23.70%
高さ: 32 Lines	16,384	4,125 26.18%	3,735 22.80%	3,492 21.31%

圧縮率高  
(b)

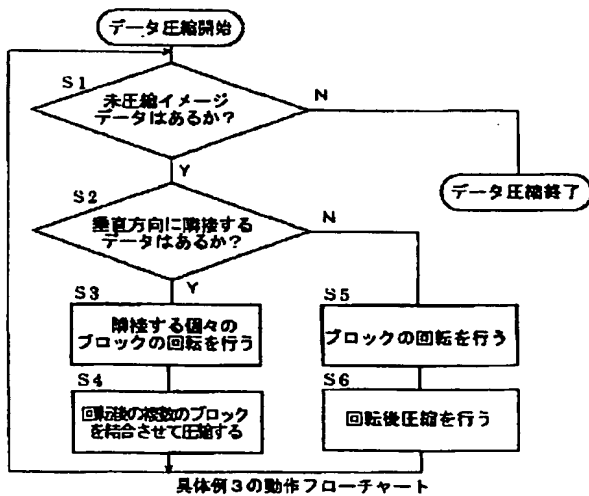
データサイズ別圧縮率比較説明図

【図 11】



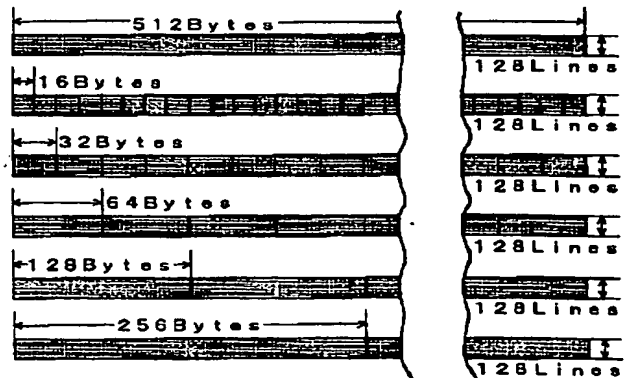
具体例3の動作説明図

【図 17】



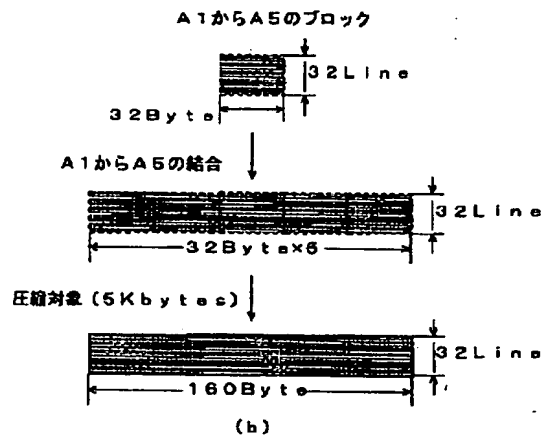
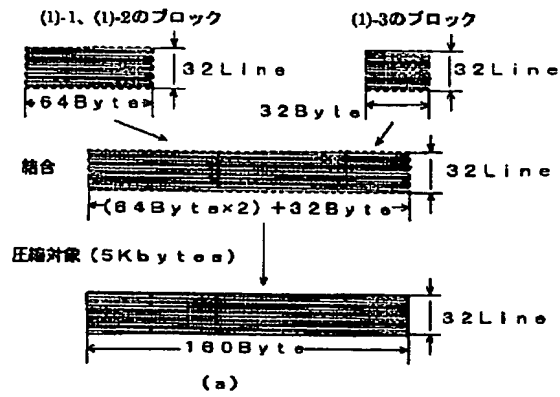
具体例3の動作フローチャート

【図 18】



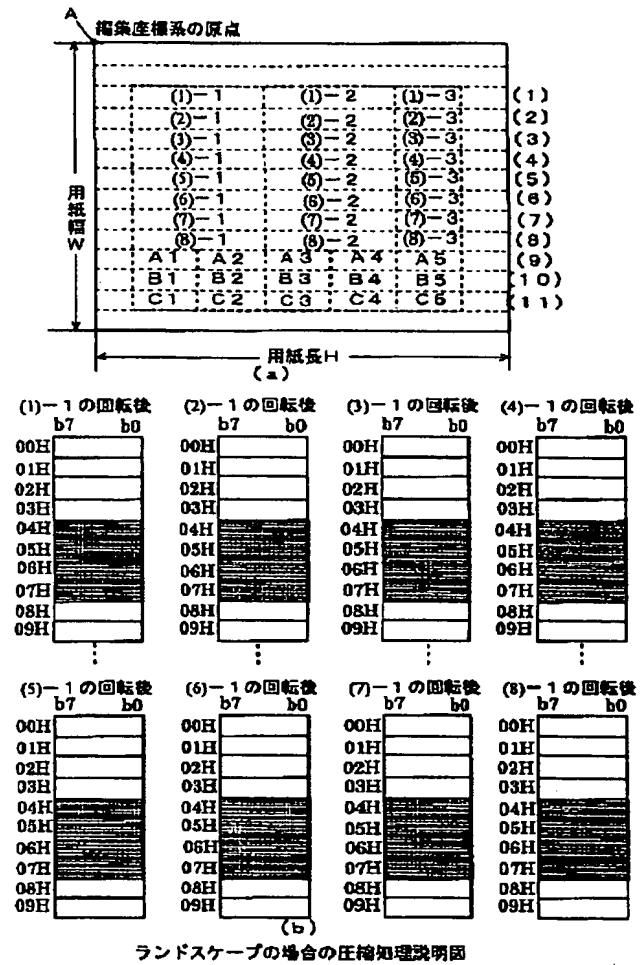
圧縮対象データの分割例説明図

【図12】

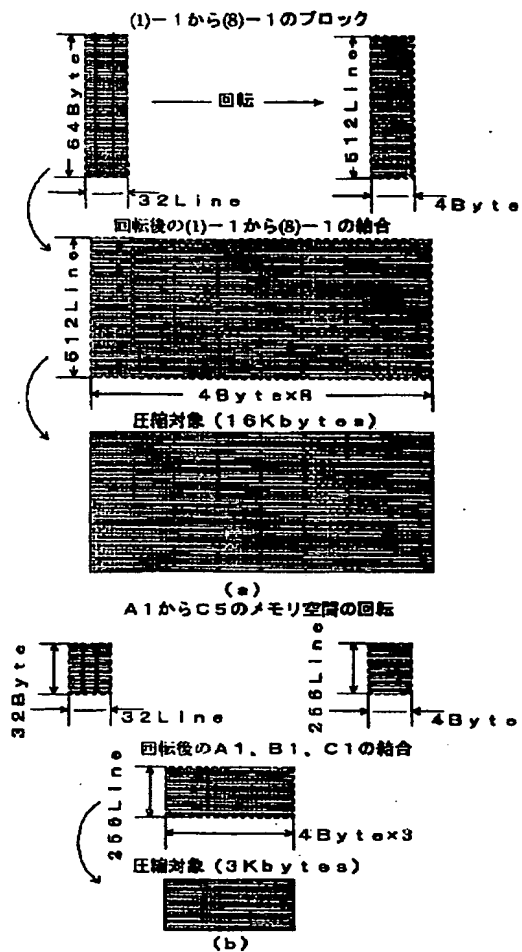


結合圧縮動作の説明図

【図15】



【図16】



フロントページの続き

Fターム(参考) 2C087 AB05 BC02 BC05 BC06 BD06  
 BD40 CA03  
 5B021 AA01 AA02 BB02 CC08 DD08  
 9A001 DD08 EE04 HH24 HH27 JJ35